UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/751,190	12/29/2000	Danny Hendler	6032-23	8523

8791   7590   10/02/2003

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD, SEVENTH FLOOR
LOS ANGELES, CA 90025

EXAMINER
PATEL, HARESH N

ART UNIT	PAPER NUMBER
2126	

DATE MAILED: 10/02/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE <u>3</u> MONTH(S) FROM
THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____ .

2a)☐ This action is **FINAL**.   2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) <u>1-27</u> is/are pending in the application.

  4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) <u>1-27</u> is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on <u>29 December 2000</u> is/are: a)☐ accepted or b)☒ objected to by the Examiner.

  Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

  If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

  a)☐ All b)☐ Some * c)☐ None of:

  1.☐ Certified copies of the priority documents have been received.

  2.☐ Certified copies of the priority documents have been received in Application No. _____ .

  3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

  * See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

  a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)   4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)   5) ☐ Notice of Informal Patent Application (PTO-152)

3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .   6) ☐ Other: .

DETAILED ACTION

1. Claims 1-27 are presented for examination.

Oath/Declaration

2. Signature of the fifth inventor (Shmuel Melamed) is missing.

Specification

Arrangement of the Specification

As provided in 37 CFR 1.77(b), the specification of a utility application should include the following sections in order. Each of the lettered items should appear in upper case, without underlining or bold type, as a section heading. If no text follows the section heading, the phrase "Not Applicable" should follow the section heading:

(a) TITLE OF THE INVENTION.

(b) CROSS-REFERENCE TO RELATED APPLICATIONS.

(c) STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT.

(d) INCORPORATION-BY-REFERENCE OF MATERIAL SUBMITTED ON A COMPACT DISC (See 37 CFR 1.52(e)(5) and MPEP 608.05. Computer program listings (37 CFR 1.96(c)), "Sequence Listings" (37 CFR 1.821(c)), and tables having more than 50 pages of text are permitted to be submitted on compact discs.) or

REFERENCE TO A "MICROFICHE APPENDIX" (See MPEP § 608.05(a). "Microfiche Appendices" were accepted by the Office until March 1, 2001.)

(e) BACKGROUND OF THE INVENTION.

(1) Field of the Invention.

(2) Description of Related Art including information disclosed under 37 CFR 1.97 and 1.98.

(f) BRIEF SUMMARY OF THE INVENTION.

(g) BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING(S).

(h) DETAILED DESCRIPTION OF THE INVENTION.

(i) CLAIM OR CLAIMS (commencing on a separate sheet).

(j) ABSTRACT OF THE DISCLOSURE (commencing on a separate sheet).

(k) SEQUENCE LISTING (See MPEP § 2424 and 37 CFR 1.821-1.825. A "Sequence Listing" is required on paper if the application discloses a nucleotide or amino acid sequence as defined in 37 CFR 1.821(a) and if the required "Sequence Listing" is not submitted as an electronic document on compact disc).

3.   The disclosure is objected to because of the following informalities:

i.   CROSS-REFERENCE TO RELATED APPLICATIONS (Copending applications) is missing.

Appropriate correction is required.

4.   The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: "System and method to stream Java Archive files from a server to a client device in sequence".

5.   Applicant is reminded of the proper content of an abstract of the disclosure.

A patent abstract is a concise statement of the technical disclosure of the patent and should include that which is new in the art to which the invention pertains. If the patent is of a basic nature, the entire technical disclosure may be new in the art, and the abstract should be directed to the entire disclosure. If the patent is in the nature of an improvement in an old apparatus, process, product, or composition, the abstract should include the technical disclosure of the improvement. In certain patents, particularly those for compounds and compositions, wherein the process for making and/or the use thereof are not obvious, the abstract should set forth a process for making and/or use thereof. If the new technical disclosure involves modifications or alternatives, the abstract should mention by way of example the preferred modification or alternative.

The abstract should not refer to purported merits or speculative applications of the invention and should not compare the invention with the prior art.

Where applicable, the abstract should include the following:

(1) if a machine or apparatus, its organization and operation;
(2) if an article, its method of making;
(3) if a chemical compound, its identity and use;
(4) if a mixture, its ingredients;
(5) if a process, the steps.

Extensive mechanical and design details of apparatus should not be given.

The abstract of the disclosure is objected to because it does not contain computer

terminology. Key terms involved in the invention like "Java" and "JAR file" are missing in the

abstract. Correction is required. See MPEP § 608.01(b).

Drawings

6.   Figures 2 and 4 should be designated by a legend such as --Prior Art-- because only that

which is old is illustrated. See MPEP § 608.02(g). A proposed drawing correction or corrected

drawings are required in reply to the Office action to avoid abandonment of the application. The

objection to the drawings will not be held in abeyance.

Information Disclosure Statement

7.   An initialed and dated copy of Applicant's IDS forms 1449 is attached to the instant

Office action.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the
subject matter which the applicant regards as his invention.

8.   Claim 2 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing

to particularly point out and distinctly claim the subject matter which applicant regards as the

invention.

The term "a ZIP file end-of-central-directory record" in claim 2 renders the claim

indefinite.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in
> section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are
> such that the subject matter as a whole would have been obvious at the time the invention was made to a person
> having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the
> manner in which the invention was made.

9.   Claims 1-51 are rejected under 35 U.S.C. 103(a) as being unpatentable over White et al.

6,230,184 (Hereinafter White).

10.   As per claims 1, 11, 20, 24 and 27, White teaches the following:

a method of streaming an archive file from a server to a client device (e.g., a client can

download a document from or send an electronic mail message to another computer using the

internet col. 1, lines 42 – 55), the method comprising,

a computer-implemented method of transmitting data in an archive file from a server

device to a client device (e.g., a request is a message from the client to the server asking for

something, such as a classfile or jar file, col 4, lines 30 – 67), the client device comprising an

execution environment configured to provide ones of a collection of logically separate files in a

received archive file to an executing application in an execution-time determined order (e.g., A

JAR file, for example, enables the user to place a collection of compressed files into a single

archive file, col. 4, lines 30 –67), the method comprising,

a data storage apparatus comprising instructions to configure a computer to,

a system for transferring information modules between computers, the system

comprising:

at a server, extracting ones of a plurality of modules from a first archive file (e.g., figure

7, Once optimized file 730 is created, an optimized version of program 700, called the optimized

program 725, is stored on server 751 in data store 775. The optimized program 725 comprises

an optimized file 730 and one or more additional files 770 (e.g. remainder files 635), col. 13

lines 1 – 16),

receiving the streamed modules at the client device (e.g., figure 7, When a client

computer issues a request to server 751 for the optimized program 725 the optimized file 730 is

sent first through the network 760. Additional files 770 may be subsequently sent to the client

computer during or after execution of the optimized file 730, col. 13, lines 1 – 16),

automatically constructing a second archive file at the client device (e.g., figure 7, When

a client computer issues a request to server 751 for the optimized program 725 the optimized file

730 is sent first through the network 760. Additional files 770 may be subsequently sent to the

client computer during or after execution of the optimized file 730, col. 13, lines 1 – 16),

the second archive file comprising the received modules (e.g., figure 7, When a client

computer issues a request to server 751 for the optimized program 725 the optimized file 730 is

sent first through the network 760. Additional files 770 may be subsequently sent to the client

computer during or after execution of the optimized file 730, col. 13, lines 1 – 16), and

providing data from at least one of the received modules in the second archive to an

executing application (e.g., figure 7, When a client computer issues a request to server 751 for

the optimized program 725 the optimized file 730 is sent first through the network 760.

Additional files 770 may be subsequently sent to the client computer during or after execution of

the optimized file 730, col. 13, lines 1 – 16),

streaming the extracted modules from the server to the client device (e.g., figure 7, When

a client computer issues a request to server 751 for the optimized program 725 the optimized file

730 is sent first through the network 760. Additional files 770 may be subsequently sent to the

client computer during or after execution of the optimized file 730, col. 13, lines 1 – 16).

However White does not specifically show the use of streaming the extracted modules

from the server to the client. "Official Notice" is taken that both the concept and advantages of

providing the streaming data to the user is well known and expected in the art.

It would have been obvious to one of ordinary skill in the art at the time the invention

was made to include streaming of the extracted modules data with the teachings of White in

order to facilitate display of animation information and/or to provide streaming audio to the user

at the client device.

11.   As per claims 2-10, 12-19, 21-23, 25 and 26, White teaches the following:

the first archive file is a Java Archive file (e.g., compressed file, col. 5, line 7 – col 6. line

58); a first one of the streamed modules comprises a ZIP file end-of-central-directory record

(e.g., zip file, col. 5, line 7 – col. 6, line 58); and the method further comprises allocating storage

space at the client device for the second archive based on size information in the end-of-central-

directory record (e.g., size of zip file, col. 5, line 7 – col. 6, line 58),

the client device comprises a Java Micro Edition execution environment (e.g., Java

runtime environment, col. 3, lines 1 – 67),

streaming comprises streaming in accordance with predetermined criteria predicting an

order of utilization of modules at the client device (e.g., desired state of computer, abstract),

a first one of the streamed modules comprises a Java class file (e.g., java applet, for

example, is comprised of classfiles, image files, sound files, data files, etc, col. 5, line 7 – col. 6,

line 58),

the executing application comprises a Java application (e.g., java applet, col. 5, line 7 –

col. 6, line 58); and providing data from at least one of the stored file to the executing application

comprises dynamically loading the Java class file for access by the Java application (e.g., Java

runtime environment, col. 3, lines 1 – 67),

at least one of the modules in the Java Archive file comprises Java Archive meta

information comprising an identification of files in other ones of the modules; and the method

further comprises streaming the meta- information from the server to the client device and

validating a file in a received modules using the meta-information (e.g., The following steps, for

example, enable the user to create a JAR file: 1) manually identify the files to be placed in the

JAR file 2) create a manifest file 3) if the files in the JAR archive are going to be digitally

signed, the appropriate digital signature files are placed in a subdirectory called META-INF 4)

combine the META-INF files with the other files and create a single ZIP file, col. 5 lines 1 – 67),

validating comprises comparing a file name and path associated with a file received in a

streamed module to a file name and path specified in the meta- information (e.g., The following

steps, for example, enable the user to create a JAR file: 1) manually identify the files to be placed

in the JAR file 2) create a manifest file 3) if the files in the JAR archive are going to be digitally

signed, the appropriate digital signature files are placed in a subdirectory called META-INF 4)

combine the META-INF files with the other files and create a single ZIP file, col. 5 lines 1 – 67),

the meta-information comprises a first digital signature and the method further

comprises: computing a second digital signature based on data in a first one of the received

modules; and comparing the second digital signature to the first digital signature to determine

validity of a file received in a streamed module (e.g., The following steps, for example, enable

the user to create a JAR file: 1) manually identify the files to be placed in the JAR file 2) create a

manifest file 3) if the files in the JAR archive are going to be digitally signed, the appropriate

digital signature files are placed in a subdirectory called META-INF 4) combine the META-INF

files with the other files and create a single ZIP file, col. 5 lines 1 – 67),

at the client device, processing a request from the executing application to access a file in

a first one of the modules: and determining if the first module is present in the second archive;

sending streaming control data to the server to request the first module when the first module is

not present in the second archive, and streaming the first module from the server to the client

device in response to receipt of said streaming control data at the server (e.g., For example, if

Java applet 200 is contained in an archive file 300 named jarFilejar the following HTML code

results in a request for the file named jarFilejar. When a request is made the file jarFilejar applet

200 is transferred from server 259 to client 251. The archive attribute can list one or more jar

files, all of which will be downloaded before the applet begins execution, col. 6 lines 1 – 67),

streaming the first module in response to the receipt of said streaming control data

comprises interrupting streaming of another one of the modules (e.g., For example, if Java applet

200 is contained in an archive file 300 named jarFilejar the following HTML code results in a

request for the file named jarFilejar. When a request is made the file jarFilejar applet 200 is

transferred from server 259 to client 251. The archive attribute can list one or more jar files, all

of which will be downloaded before the applet begins execution, col. 6 lines 1 – 67),

the second archive file comprises a different arrangement of the streamed files than in the

first file, said different arrangement being functionally equivalent to an arrangement of files in

the first archive file (e.g., For example, if Java applet 200 is contained in an archive file 300

named jarFilejar the following HTML code results in a request for the file named jarFilejar.

When a request is made the file jarFilejar applet 200 is transferred from server 259 to client 251.

The archive attribute can list one or more jar files, all of which will be downloaded before the

applet begins execution, col. 6 lines 1 – 67),

the first archive comprises at least one file comprising non-executable data and at least

one file comprising executable program code (e.g., an applet or application, col. 6, lines 1-67),

ordering the separate files extracted from the archive file based on criteria predicting an

order of utilization of the separate files at the second computing device; and streaming the

separate files from the first computer to the second computer in accordance with the ordering,

the criteria comprises data stored at the first computer in a streaming control database;

and the method further comprises: receiving a request at the first computer from the second

computer; and modifying the predictive data stored in the streaming control database based on

the request,

the criteria comprise data stored at the first computer in a streaming control database

comprising transition records associating weighted values with execution transitions between

selected files in the collection; and ordering the separate files comprises ordering based on

transition record values,

ordering based on transition record values comprises processing transition record

information using a path determination algorithm (e.g., This command generates a compressed

JAR file. The jar command also generates a default manifest file called MANIFEST.MF and

places it in a directory called META-INF. The manifest file is a file that contains information

about the files packaged in the JAR file. The c option indicates that the user wants to create a

JAR file. The f option indicates that the output is sent to a file rather than to an output stream.

The c and the f option can appear in any order, but should not be separated by a space. The jar-

file argument is the name of the JAR file the user wants to create. The input-file(s) argument is a

space-delimited list of one or more files the user wants to place in the JAR file. The input-file(s)

argument can contain a wildcard (*) symbol. If any of the input-file(s) are directories, the

contents of these directories are added to the JAR file recursively, col. 5, lines 29 – 43),

receiving a first module in the module set while an application is executing; storing the

first module on local storage media at the second computer; and wherein the method further

comprises integrating the first module with the executing application (e.g., This command

generates a compressed JAR file. The jar command also generates a default manifest file called

MANIFEST.MF and places it in a directory called META-INF. The manifest file is a file that

contains information about the files packaged in the JAR file. The c option indicates that the

user wants to create a JAR file. The f option indicates that the output is sent to a file rather than

to an output stream. The c and the f option can appear in any order, but should not be separated

by a space. The jar-file argument is the name of the JAR file the user wants to create. The

input-file(s) argument is a space-delimited list of one or more files the user wants to place in the

JAR file. The input-file(s) argument can contain a wildcard (*) symbol. If any of the input-

file(s) are directories, the contents of these directories are added to the JAR file recursively, col.

5, lines 29 – 43),

the instructions to construct in accordance with ZIP file format specifications further

comprise instructions to position received ones of the modules in the second archive in an order

of receipt of the modules and instructions to alter data in the second archive's central directory

header to indicate the placement of the ones of the modules in the second archive (e.g., This

command generates a compressed JAR file. The jar command also generates a default manifest

file called MANIFEST.MF and places it in a directory called META-INF. The manifest file is a

file that contains information about the files packaged in the JAR file. The c option indicates

that the user wants to create a JAR file. The f option indicates that the output is sent to a file

rather than to an output stream. The c and the f option can appear in any order, but should not be

separated by a space. The jar-file argument is the name of the JAR file the user wants to create.

The input-file(s) argument is a space-delimited list of one or more files the user wants to place in

the JAR file. The input-file(s) argument can contain a wildcard (*) symbol. If any of the input-

file(s) are directories, the contents of these directories are added to the JAR file recursively, col.

5, lines 29 – 43),

the instructions further comprise instructions to: process a request from an executing

application to access data in a first one of the modules; determine if the first module is present in

the second archive; send control data to the other computer to request the first module when the

first module is not present in the second archive (e.g., This command generates a compressed

JAR file. The jar command also generates a default manifest file called MANIFEST.MF and

places it in a directory called META-INF. The manifest file is a file that contains information

about the files packaged in the JAR file. The c option indicates that the user wants to create a

JAR file. The f option indicates that the output is sent to a file rather than to an output stream.

The c and the f option can appear in any order, but should not be separated by a space. The jar-

file argument is the name of the JAR file the user wants to create. The input-file(s) argument is a

space-delimited list of one or more files the user wants to place in the JAR file. The input-file(s)

argument can contain a wildcard (*) symbol. If any of the input-file(s) are directories, the

contents of these directories are added to the JAR file recursively, col. 5, lines 29 – 43),

instructions to order the extracted modules based on criteria predicting an order of

utilization of the modules at the client device and wherein the instructions to stream comprise

instructions to stream in accordance with the ordering of the modules (e.g., This command

generates a compressed JAR file. The jar command also generates a default manifest file called

MANIFEST.MF and places it in a directory called META-INF. The manifest file is a file that

contains information about the files packaged in the JAR file. The c option indicates that the

user wants to create a JAR file. The f option indicates that the output is sent to a file rather than

to an output stream. The c and the f option can appear in any order, but should not be separated

by a space. The jar-file argument is the name of the JAR file the user wants to create. The

input-file(s) argument is a space-delimited list of one or more files the user wants to place in the JAR file. The input-file(s) argument can contain a wildcard (*) symbol. If any of the input-file(s) are directories, the contents of these directories are added to the JAR file recursively, col. 5, lines 29 – 43),

the instructions to order the extracted modules comprise instructions to order in accordance with criteria retrieved from a streaming control database (e.g., This command generates a compressed JAR file. The jar command also generates a default manifest file called MANIFEST.MF and places it in a directory called META-INF. The manifest file is a file that contains information about the files packaged in the JAR file. The c option indicates that the user wants to create a JAR file. The f option indicates that the output is sent to a file rather than to an output stream. The c and the f option can appear in any order, but should not be separated by a space. The jar-file argument is the name of the JAR file the user wants to create. The input-file(s) argument is a space-delimited list of one or more files the user wants to place in the JAR file. The input-file(s) argument can contain a wildcard (*) symbol. If any of the input-file(s) are directories, the contents of these directories are added to the JAR file recursively, col. 5, lines 29 – 43), and

modify the criteria stored in the streaming control database based on a module request received from the second computer (e.g., This command generates a compressed JAR file. The jar command also generates a default manifest file called MANIFEST.MF and places it in a directory called META-INF. The manifest file is a file that contains information about the files packaged in the JAR file. The c option indicates that the user wants to create a JAR file. The f option indicates that the output is sent to a file rather than to an output stream. The c and the f

option can appear in any order, but should not be separated by a space. The jar-file argument is

the name of the JAR file the user wants to create. The input-file(s) argument is a space-delimited

list of one or more files the user wants to place in the JAR file. The input-file(s) argument can

contain a wildcard (*) symbol. If any of the input-file(s) are directories, the contents of these

directories are added to the JAR file recursively, col. 5, lines 29 – 43),

Conclusion

12.   This application is a continuation in part of application number 09/120,575, which does

not teach the entire claimed invention.

13.   The prior art made of record and not relied upon is considered pertinent to applicant's

disclosure.

See Form PTO-892.

Getchius et al. (6,408, 294) clearly teaches a system for performing online data queries

that displays advertisements based on the client request terms for the business listing. The

machine executable code identifies terms that are related to one or more of client terms

associated with a business listing. Getchius teaches web-based systems for streaming additional

advertisement web site elements that are related to the client request.

Barry (6, 615,258) teaches an integrated data management system for providing data

management services from an enterprise over the Internet. Barry teaches mechanism to decrease

the user perceived system response time in web-based systems. Barry teaches usage streaming of

JAR files from a server to a client device.

Jones (6,256,623) teaches the portal host system containing search clip records and one

or more tags identifying search words or topics that the clip is related to, to identify related

topics.

Forbes et al. (6,381,742) also teaches streaming of JAR files from a server to a client

device.

Any inquiry concerning this communication or earlier communications from the

examiner should be directed to Haresh Patel whose telephone number is (703) 605-5234. The

examiner can normally be reached on Monday-Friday from 8:00 am to 5:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's

supervisor, John Follansbee, can be reached at (703) 305-8498.

The appropriate fax phone number for the organization where this application or

proceeding is assigned is (703) 306-5404.

Any inquiry of a general nature or relating to the status of this application or proceeding

should be directed to the receptionist whose telephone number is (703) 305-3900.

Haresh Patel

September 17, 2003.

JOHN FOLLANSBEE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100